USHER: Holistic Interference Avoidance for Resource Optimized ML Inference

Paper Information

- Title: USHER: Holistic Interference Avoidance for Resource Optimized ML Inference
- Authors: Sudipta Saha Shubha, Haiying Shen (University of Virginia), Anand Iyer (Georgia Institute of Technology)
- Conference: Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2024)
- Date: July 10–12, 2024

Research Background

随着深度学习模型的普及和模型尺寸的快速增长，ML 推理已成为生产环境中的主要成本之一。现有的 ML 推理系统虽然通过多种优化手段提高了 GPU 的利用率，但模型间的资源竞争导致的推理延迟增加和 SLO（服务水平目标）违规问题仍然存在。因此，本文提出了一种新的系统 USHER，以全局的视角避免干扰，优化资源利用。

Introduction and Motivation

论文首先介绍了当前 ML 推理服务面临的挑战，包括模型尺寸的快速增长、GPU 资源的昂贵和功率消耗大等。然后，回顾了现有的 ML 推理系统，如 Shepherd、GPUlet 和 AlpaServe 等，分析了它们在提高 GPU 利用率和避免模型间干扰方面的优缺点。

Observation1：现有的推理服务系统无法最大化 Cuti 或 Muti，且模型复用会因模型间的干扰而显著降低吞吐量，同时最大化 Cuti 并不一定意味着能最大化 Muti。

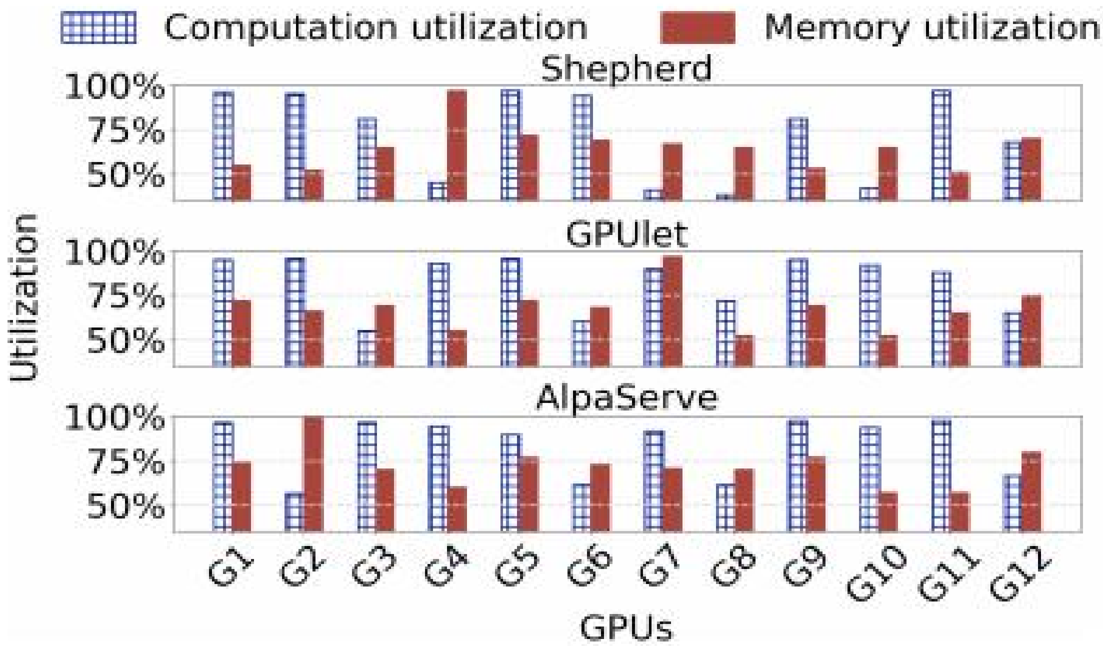


Figure 1: Performance of existing systems.

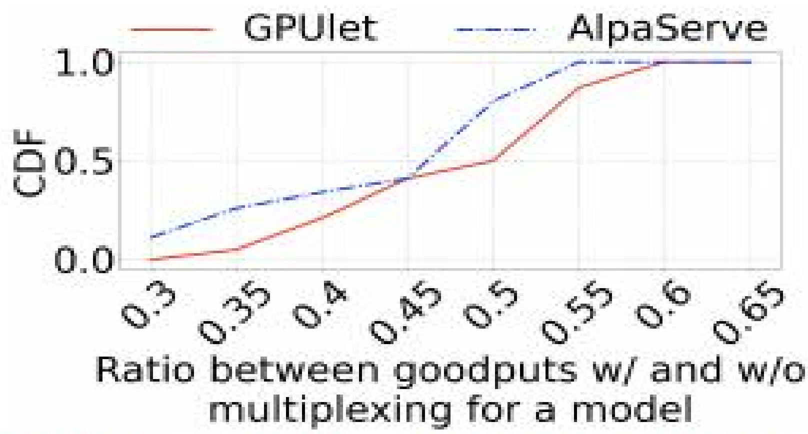


Figure 2: Impact of inter-model interference on goodput.

Observation2：在基于空间复用的推理服务中，与现有系统不同，即使一个 GPU 就足够在服务水平目标（SLO）内完成工作量，我们可能仍然需要拆分模型的工作量，以提高整体资源利用率。

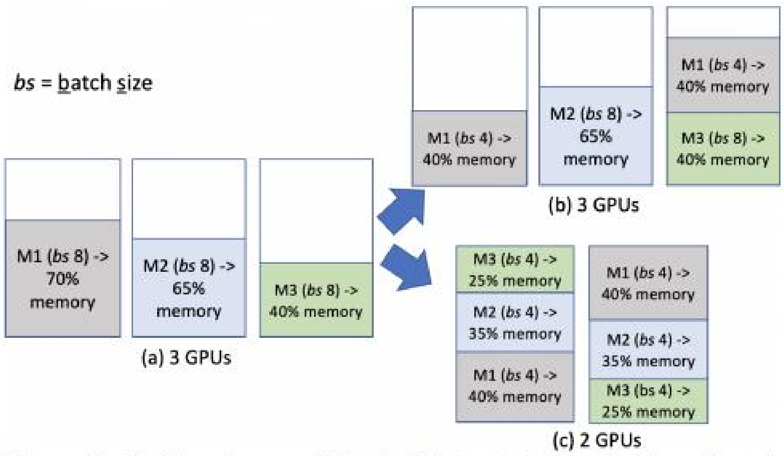
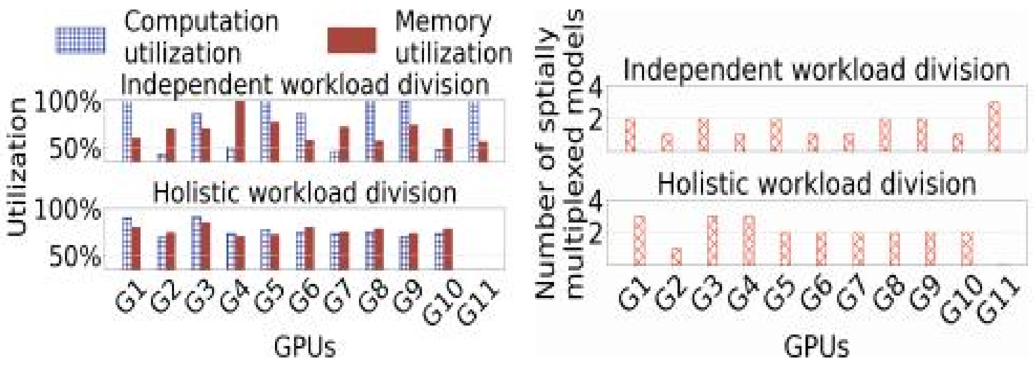


Figure 3: Performing workload division holistically for all models increases resource utilization.

Observation3：每个模型的工作量分配不应单独决定。相反，必须采用一种同时考虑所有模型的整体方法。



(a) Resource utilization of each GPU by workload divison.

(b) Number of spatially multiplexed models in each GPU using workload division.

Figure 4: Effectiveness of holistic workload division.

Observation4：与普遍认知不同，仅凭模型参数大小并不能决定一个模型是 C 密集型还是 M 密集型。即使是一个小模型，在 C 需求（Creq）方面也可能超越一个更大的模型，这取决于批处理大小（BS）、与 BS 相关的资源需求和服务水平目标（SLO）之间的复杂关系。

Observation5：将 C 密集型模型与 M 密集型模型进行复用可以提高 GPU 的 Cuti 和 Muti。

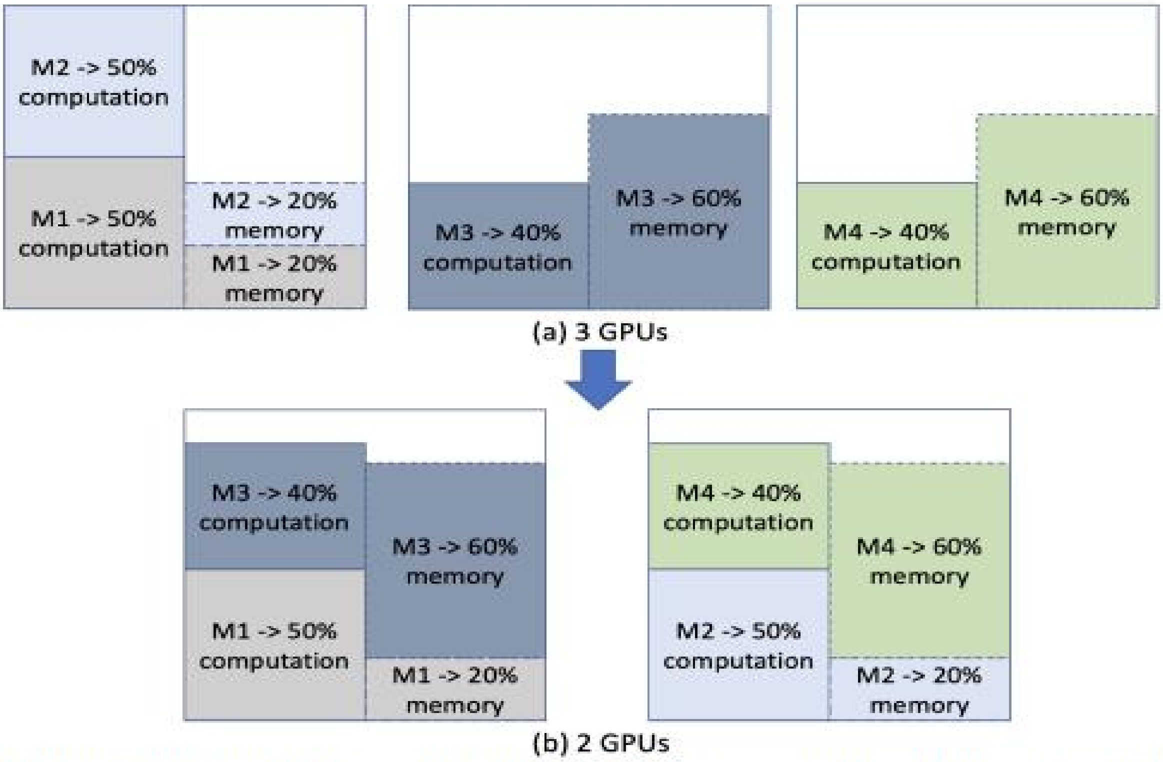
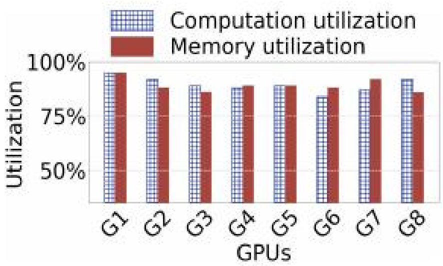
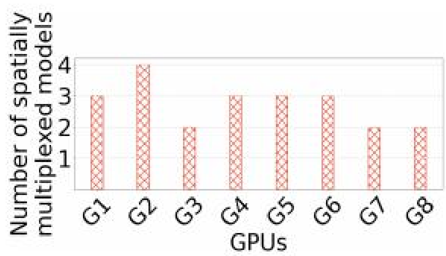


Figure 7: Spatially multiplexing a computation-heavy model with a memory-heavy model increases resource utilization.



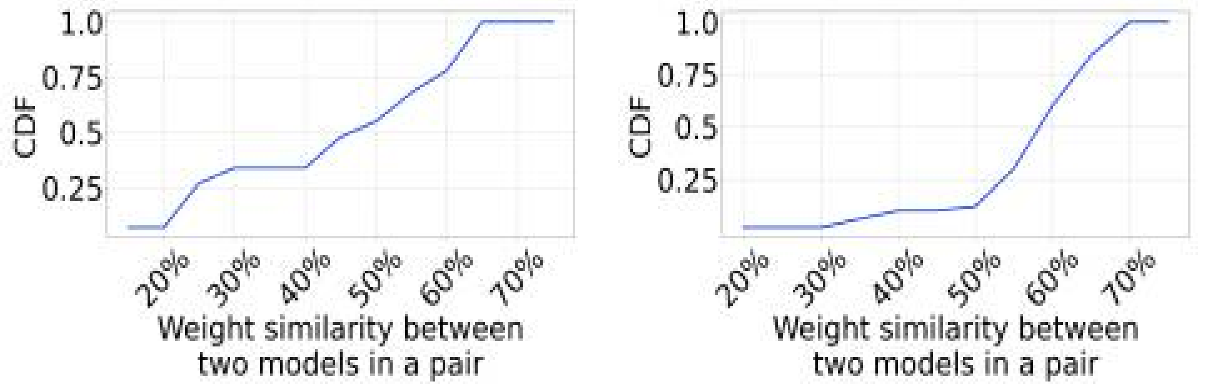
(b) Resource utilization by multiplexing C-heavy and M-heavy models.



(c) Model# by multiplexing C-heavy and M-heavy models.

与图 4a 中的整体方法相比，平均 Cuti 和 Muti 分别提高了 12.1%和 11.8%。此外，与图 4b 中的整体方法相比，平均模型数量（model#）增加了 0.6。

Observation6：不同卷积神经网络（CNN）模型之间以及不同 Transformer 模型之间都存在显著的权重重叠。



(a) Weight similarity across CNN models. **(b) Weight similarity across Transformer models.**

Figure 8: Weight similarity across models.

System Design

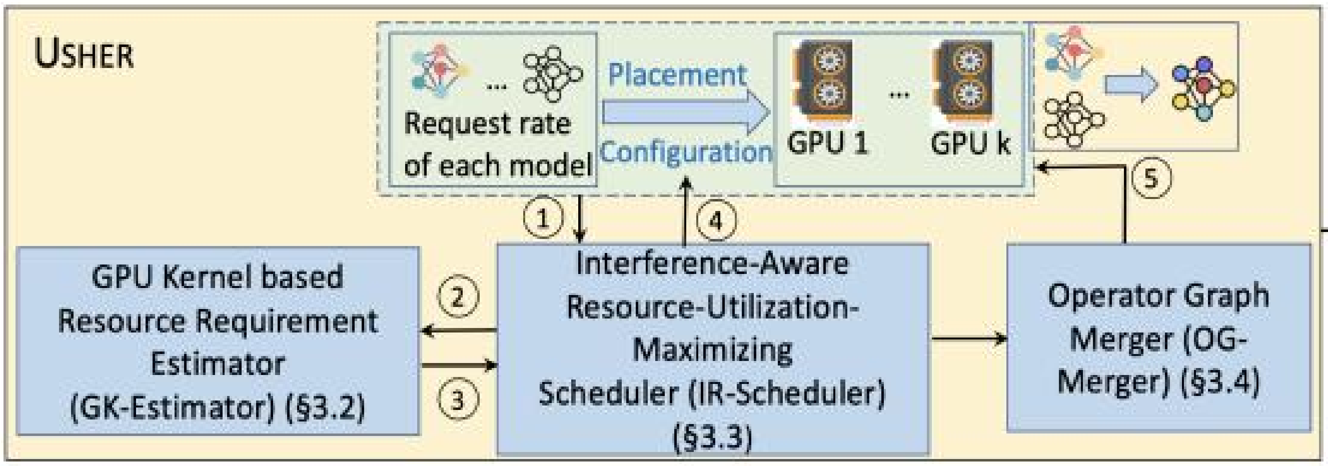
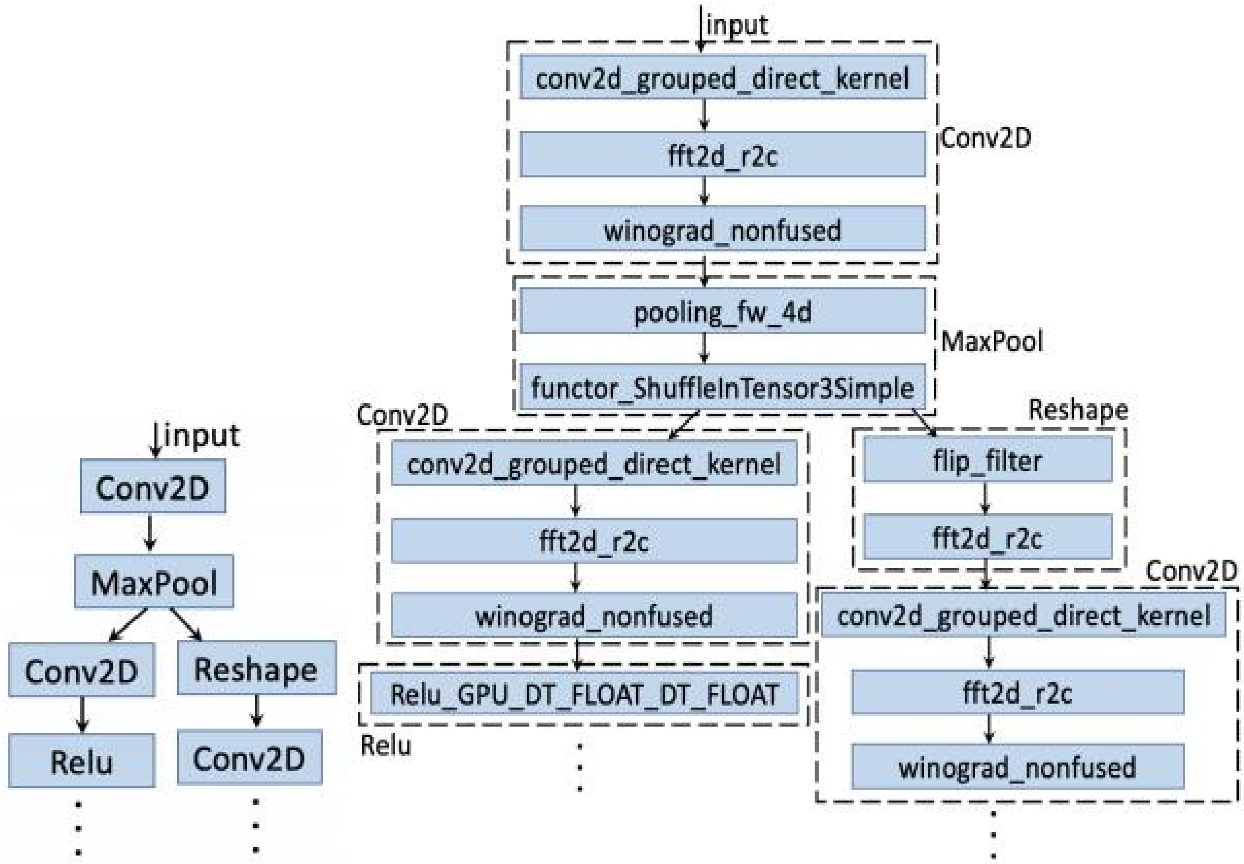


Figure 9: System overview of USHER.

USHER 系统的设计主要包括三个部分：基于 GPU 内核的资源需求估计器、启发式干扰感知资源利用最大化调度器和新型操作符图合并方法。

1. 基于 GPU 内核的资源需求估计器：通过分析模型将执行的 GPU 内核，估计其资源需求，避免了成本高昂且耗时的离线剖析。



(a) Operator-level computation graph (b) Kernel-level computation graph

Figure 10: Conversion of an operator-level computation graph for a CNN to its kernel-level computation graph.

2. 启发式干扰感知资源利用最大化调度器：根据资源需求估计，决定每个模型的批处理大小、模型复制程度和模型放置位置，以最小化货币成本并满足延迟 SLO 或最大化吞吐量。

Step1：Model Grouping

USHER 首先确定每个模型的 C 需求（Creq）和 M 需求（Mreq）。然后，USHER 使用 GK-Estimator 计算所有可能的批处理大小（BS）和 GPU 类型组合下的平均 R 需求（Rreq）。接下来，USHER 使用 k-means 聚类算法的一种变体进行分组。

Step2: Scheduling

Algorithm 1 Interference-aware and resource utilization-maximizing scheduler for **G**.

for each $\mathcal{G}_i \in \mathbf{G}$ **do**
  Generate all possible `configurations={(BS, RD)` for each model $M \in \mathcal{G}_i$}.
  for each `configuration` **do**:
    `cost`, `total_goodput` = `PLACEMENT(configuration)`
  Schedule as per the `configuration` for which all of the requests are completed within their latency SLOs, i.e., `total_goodput` = `total_workload` and the `cost` is minimum.

Algorithm 2 Placement algorithm for model group $\mathcal{G}_i$.

procedure `PLACEMENT(configuration)`
  $\mathcal{G}_{iGPU} \leftarrow$ GPU group for $\mathcal{G}_i$, initially empty
  for each $M \in \mathcal{G}_i$ **do**
    Calculate its $Creq$ and $Mreq$ in each type of GPU
    if $Creq > \max C$ or $Mreq > \max M$ (highest-capacity GPU) **then**
      return `Infeasible_configuration`
  Group the models into C-heavy and M-heavy models
  Sort two groups in descending order of $Creq + Mreq$:
  $\{M_1, M_2, \ldots, M_n\}$ and $\{M'_1, M'_2, \ldots, M'_m\}$
  `final_model_list` $\leftarrow \{(M_1, M'_1), (M_2, M'_2), \ldots, (M_n, M'_m)\}$
  for each $M \in$ `final_model_list` **do**
    `MODEL_REPLICA_PLACEMENT_WITHIN_`$\mathcal{G}_{iGPU}$`()`
    `MODEL_REPLICA_PLACEMENT_OUTSIDE_`$\mathcal{G}_{iGPU}$`()`
    for each model replica of M **do**
      `NEW_LOWEST_COST_GPU_INITIALIZATION()`
      Assign the new GPU to $\mathcal{G}_{iGPU}$.
  for each $M \in \mathcal{G}_i$ **do**
    $\texttt{goodput}_M = \texttt{min}(\texttt{achieved\_goodput}_M, \texttt{workload}_M)$
  $\texttt{total\_goodput} = \sum_M \texttt{goodput}_M$
  return additional `costs` for initializing new GPUs and `total_goodput` for the taken placement decision.

3. 新型操作符图合并方法：在分配给同一 GPU 的模型间，尽可能合并操作符图，以减少 GPU 缓存中的干扰。

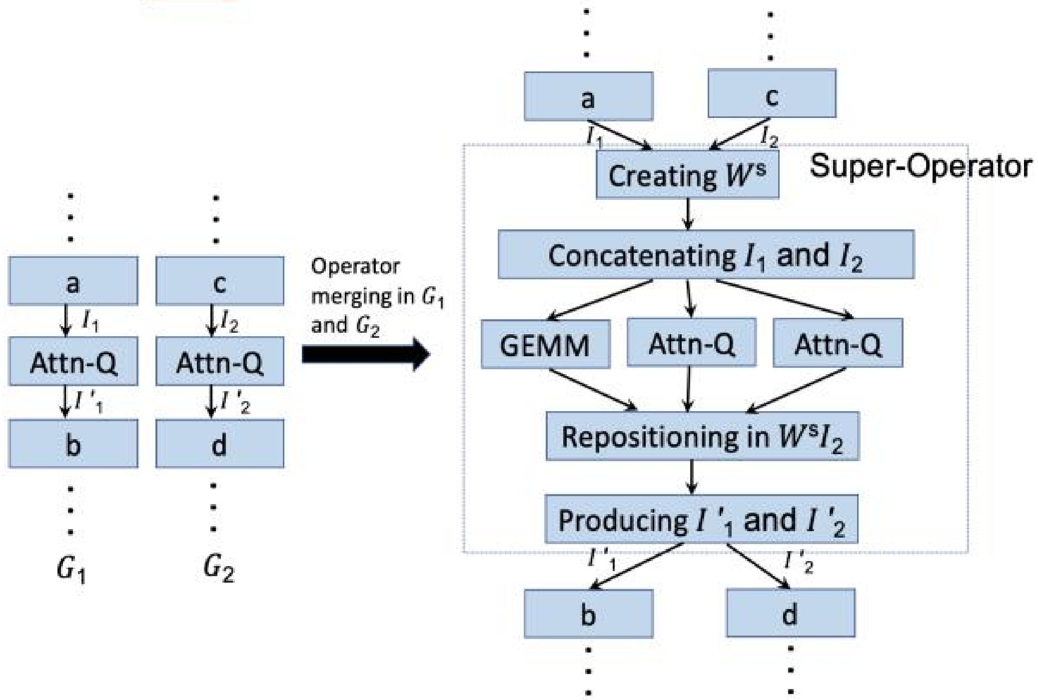
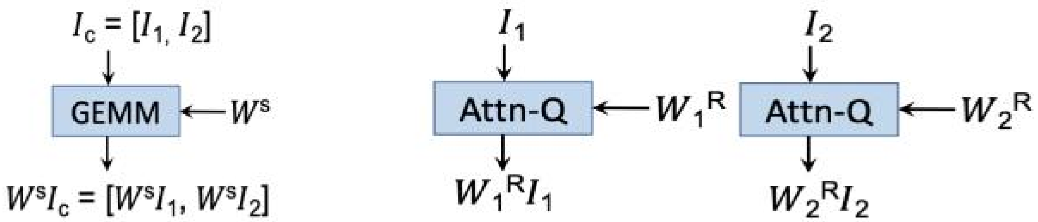


Figure 11: Operator merging in G_1 and G_2 to maximize GPU cache usage. Attn-Q refers to the Attention Query operator.



(a) Executing on W^s at the same time.

(b) Executing the remaining operations using separate operators.

Figure 12: Creating a new operator GEMM.

Implementation Details

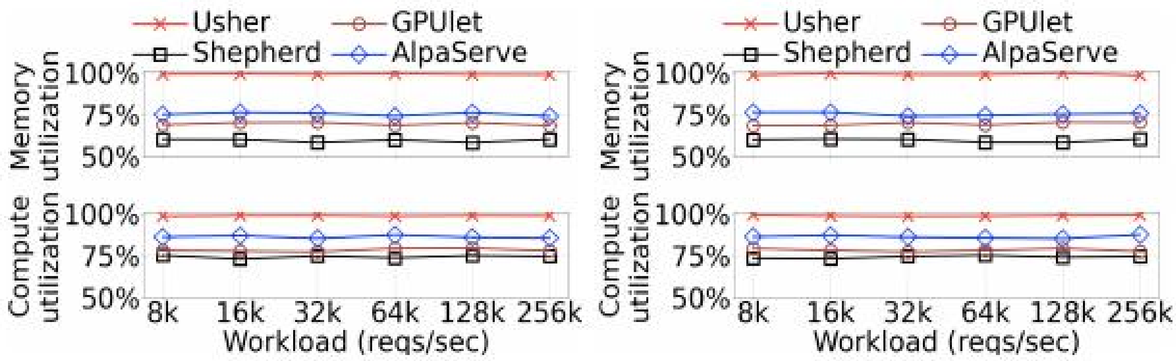
该部分提供了有关 USHER 如何开发和实施的有用信息。作者提到了使用 Python 和 TensorFlow，以及将操作符图转换为 ONNX 格式以确保与其他机器学习框架的兼容性。此外，还提到了使用 Nvidia Nsight 进行性能分析和使用 Nvidia MPS 进行资源划分，这证明了所提出系统的实用性。

Experimental Evaluation

实验评估部分严谨且全面。作者在真实测试平台和大规模模拟环境中进行了实验，将 USHER 与 GPUlet 和 AlpaServe 等现有系统进行了比较。实验结果表明，USHER 在吞吐量和成本效率方面均有显著提升，支持了论文中的论点。

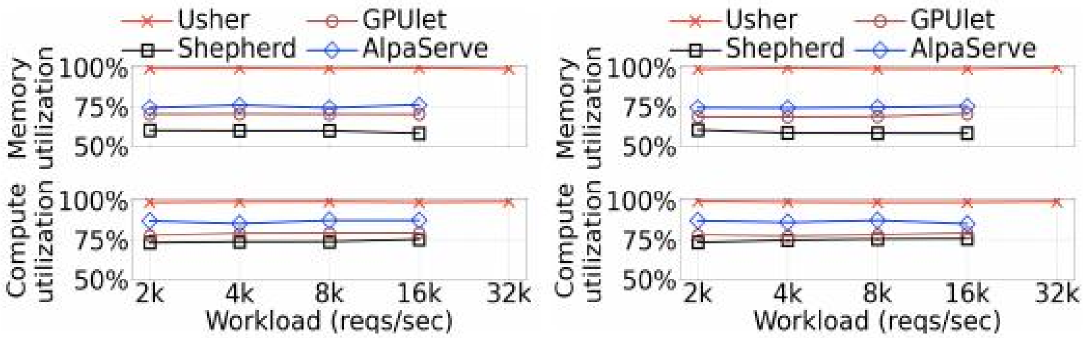
(a) MAF1 (b) MAF2

Figure 13: Goodput comparison of different methods in real testbed for a fixed cluster.



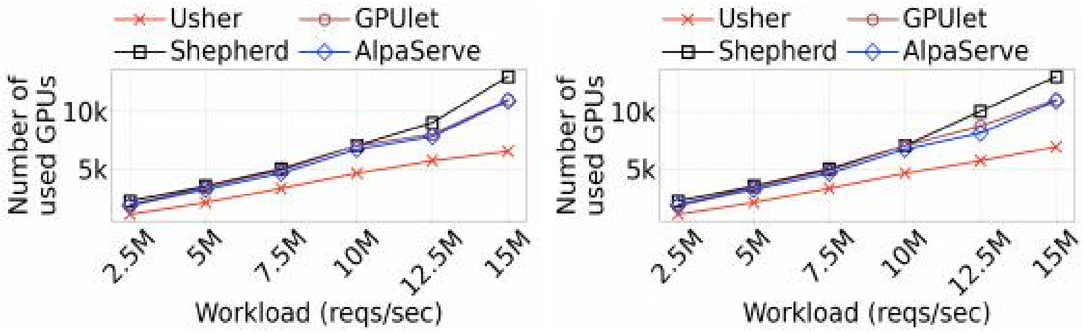
(a) MAF1 (b) MAF2

Figure 14: GPU computation and memory utilization comparison of different methods in real testbed for a fixed cluster.



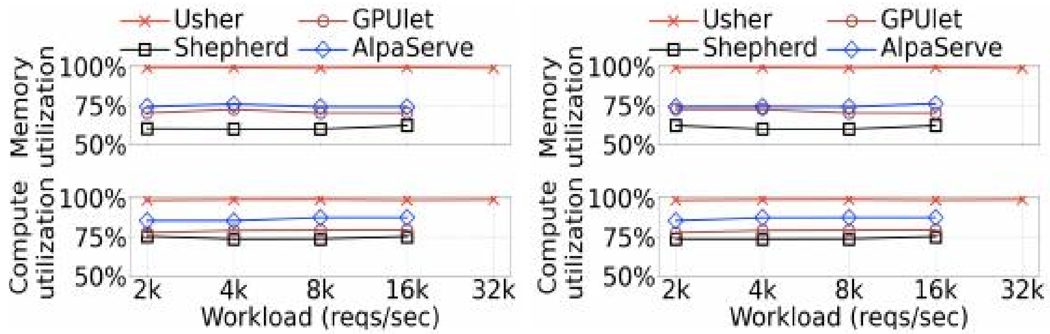
(a) MAF1 **(b) MAF2**

Figure 17: GPU computation and memory utilization of different methods in real testbed for a homogeneous non-fixed cluster.



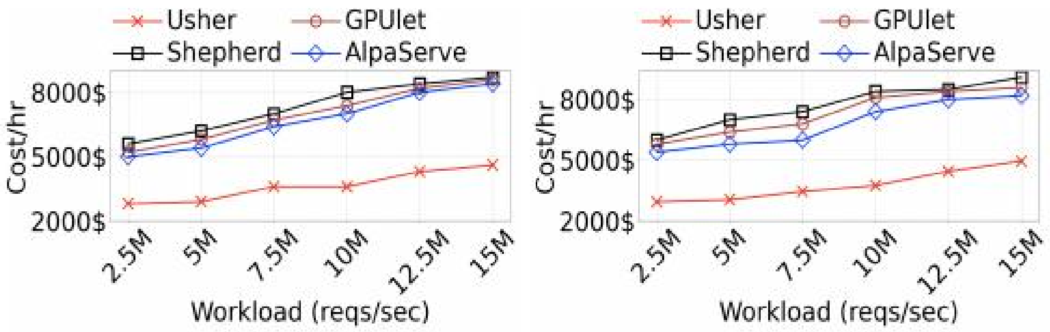
(a) MAF1 **(b) MAF2**

Figure 18: Number of used GPUs of different methods in simulation for a homogeneous non-fixed cluster.



(a) MAF1 **(b) MAF2**

Figure 20: GPU computation and memory utilization comparison in real testbed for a heterogeneous non-fixed cluster.



(a) MAF1 **(b) MAF2**

Figure 21: Cost comparison of different methods in simulation for a heterogeneous non-fixed cluster.

Discussion and Limitations

该部分深入探讨了 USHER 的性能及其局限性。作者承认了参数调整所面临的挑战以及精度量化对系统性能的影响。他们还提出了未来工作的方向，例如探索使用各种精度量化，并将 USHER 扩展到能够自适应地为每个模型选择最合适的精度量化的程度。

Conclusion

论文总结指出，USHER 系统通过空间复用 GPU 资源，并在感知干扰的情况下优化计算和内存利用率，实现了资源优化 ML 推理。实验结果表明，USHER 在吞吐量和成本效率方面优于现有系统。未来工作将探索不同精度量化对系统性能的影响，并进一步扩展 USHER 的适应性。